A combination of particle filtering and deterministic approaches for multiple kernel tracking

Céline Teulière, Eric Marchand, Laurent Eck

***Abstract*— Color-based tracking methods have proved to be efficient for their robustness qualities. The drawback of such global representation of an object is the lack of information on its spatial configuration, making difficult the tracking of more complex motions. This issue can be overcome by using several kernels weighting pixels locations.**

In this paper a multiple kernels configuration is proposed and developed in both probabilistic and deterministic frameworks. The advantages of both approaches are combined to design a robust tracker allowing to track location, size and orientation of the object.

A target tracking scheme using visual servoing considering measurements provided by the presented approach validates the proposed method.

I. INTRODUCTION

Real-time object tracking is a key task required in various vision-based applications, such as visual servoing, surveillance, augmented reality, etc. The principle of object tracking through image frames is usually to determine what part of the image best corresponds to a reference model of the object's appearance.

Different kinds of descriptors can be used to represent the object of interest, depending on the task to be achieved. When the object to track is well-known and the variations of its appearance during motion are small, one can use spatial descriptors such as image templates [1] [2] [3]. Describing how the object looks like pixel-wise, image templates can recover a large range of motions. Since they are very sensitive to modifications in the object appearance (occlusions, lighting variations, etc.) they would fail in sequences where large appearance changes or partial occlusions can occur. On the contrary, density-based descriptors such as color histograms have proved to be more robust and versatile and are thus an attractive choice to deal with complex tracking tasks.

Color-based tracking algorithms have been mainly considered in two different approaches. The mean shift technique [4] consists in minimizing a distance between weighted color histograms in a gradient-based descent minimization, leading to good accuracy in tracking. However, the search being deterministic, the algorithm can fail in case of total occlusion, presence of another object of similar color, or large displacements. Within another framework, the same kind of likelihood criteria is used in [5] and [6], but embedded in a particle filter. The posterior density of probability of object location is discretized in a set of weighted particles, mean shift optimization being replaced in this context by the particles set evolution and its weighted mean. The use of a probabilistic framework provides better robustness w.r.t. occlusions or presence of similar objects. However, in real-time applications a balance has to be found between computational time and accuracy since the former is directly linked with the number of particles used.

In the studies above, the use of density-based descriptors such as color histograms allows to achieve robust localization. The resulting drawback of choosing this form of representation is the loss of spatial information on the object, making difficult to track more complex motions.

Kernel-based methods, e.g. in [4], add some spatial information by giving a higher weight to pixels close to the center of the object, but these symmetric kernels are not sensitive to, for example, rotational motions. The issue of using kernel-based descriptors to track complex motions is addressed in [8]. Linking kernel-based methods with optimization techniques, the authors show how to adapt them to multiple kernels. This allows to design trackers more sensitive to specific motions. Examples of kernels are given although no generic configuration is proposed to address a large range of motions. [9] formulates the problem in terms of inverse composition.

Within the probabilistic framework, a likelihood function, sensitive to such complex motion, has to be designed. The idea of dividing the object in several parts has been introduced in [5] to improve tracking accuracy.

In this paper we propose a multiple kernel configuration and give evidences of its improved sensitivity over single kernel approaches. The two approaches above are developed using this configuration and their advantages and comparative results are presented. A combination of both approaches is discussed. The experiments presented are restricted to tracking the position (coordinates of the center of gravity) orientation and size of an object, but the approach is general enough to be adapted to homographic motions. The multiple kernel algorithm proposed is used within a visual servoing task.

The remainder of this paper is organized as follows. Section 2 formulates color-based tracking problem and describes the proposed multiple kernels representation. This configuration is presented in a deterministic approach in section 3, and in the probabilistic framework of particle filtering in section 4. A combination of these two approaches is then proposed in

C. Teulière and L. Eck are with CEA, LIST, Service de Robotique Interactive, 18 route du Panorama, BP6, Fontenay aux Roses, F- 92265 France `firstname.name@cea.fr`, E. Marchand is with INRIA Rennes-Bretagne Atlantique, IRISA, Lagadic Project, Rennes, France `firstname.name@irisa.fr`. This work was realized in the context of the French ANR national project SCUAV (ANR Psirob 06_174032 SCUAV project ref ANR-06-ROBO-0007-02)

section 5. Section 6 shows how the tracking information can be used in a visual servoing application. Finally, experimental results are presented.

II. Color-based tracking

A. State space

The goal of any visual tracking algorithm is to estimate the state of an object or a region of interest through frames. This state represents the location, but can also contain other parameters of the target deformation.

Since density-based descriptors such as color histograms give little information on the object's spatial configuration, they are mostly used to track only location and scale. This paper deals with adapting this measure to track more complex motions. The experiments presented here, consider translations, scale and rotation modifications, allowing to use the resulting tracking in a visual servoing application with four degrees of freedom. Nevertheless, the developed methods can be used for a larger range of motions, like affine or homographic displacements.

In the remainder of this paper, without loss of generality, the considered region of interest is a rectangle of fixed ratio $r = \frac{h}{w}$, which state is then defined by its position (x, y), orientation θ and area $A = hw$ (see Figure 1). The state $\mathbf{x}_k$ of the object in frame k is then given by $\mathbf{x}_k = (x_k, y_k, \theta_k, \frac{1}{\sqrt{A_k}})^\top$.

In the following, $\{\boldsymbol{l}_i\}_{i=0..n}$ represent the initial pixels locations of the object, that is assumed to be selected or automatically detected in frame 0. If $f_{\mathbf{x}}$ denotes the affine deformation between the initial state of the object and the state $\mathbf{x}$, for each pixel location $\boldsymbol{l}_i$:

$$f_{\mathbf{x}}(\boldsymbol{l}_i) = f_{\mathbf{x}}(\mathbf{c}) + s\mathbf{R}_\theta(\boldsymbol{l}_i - \mathbf{c}) \tag{1}$$

where $\mathbf{R}_\theta$ is the 2×2 rotation matrix of angle $\theta - \theta_0$, $s = \sqrt{\frac{A}{A_0}}$ and $\mathbf{c} = (x, y)^\top$ is the center of the rectangle.

B. Measurement space

In color-based tracking, the reference object is usually characterized by its color histogram $\mathbf{q}^* = \{q_u^*\}_{u=1...m}$, m being the number of bins in which color space is divided. In the experiments RGB color space with $8 \times 8 \times 8$ bins has been used. HSV color space is another classical choice used for example in [11]. For each bin u:

$$q_u^* = \sum_{i=1}^{n} K(\boldsymbol{l}_i - \mathbf{c})\delta_u(b(\boldsymbol{l}_i)) \tag{2}$$

where $b(\boldsymbol{l}_i)$ is the bin corresponding to the color of pixel $\boldsymbol{l}_i$, δ designates Kronecker's function, and K is a kernel function centered in $\mathbf{c}$, weighting image locations. Unless explicitly mentioned K is supposed normalized so that $\sum_{u=1}^{m} q_u^* = 1$.

Whatever the tracking framework used (particle filtering, or kernel-based optimization), in each frame k, the aim is to find the candidate state $\mathbf{x}_k$ which histogram $\mathbf{q}(\mathbf{x}_k)$ is the "closest" to the reference histogram $\mathbf{q}^*$. To achieve this, a correlation criterion in the histogram space is given by Bhattacharyya coefficient:

$$\rho(\mathbf{x}_k) = \rho(\mathbf{q}^*, \mathbf{q}(\mathbf{x}_k)) = \sum_{u=1}^{m} \sqrt{q_u^* q_u(\mathbf{x}_k)} \tag{3}$$

$q_u(\mathbf{x}_k)$ being computed as in (2) with the pixels locations corresponding to the target in $\mathbf{x}_k$ *i.e.* $\{f_{\mathbf{x}_k}(\boldsymbol{l}_i)\}_{i=0..n}$.

A candidate state $\mathbf{x}_k$ for the object in frame k is then compared to the reference object using Bhattacharyya distance:

$$d(\mathbf{x}_k) = d(\mathbf{q}^*, \mathbf{q}(\mathbf{x}_k)) = \sqrt{1 - \rho(\mathbf{x}_k)} \tag{4}$$

Although this representation by a single histogram is sufficient for tracking the location and/or scale of the object, it is very little sensitive to rotation (see Figure 3), especially when the ratio r is close to 1.

The next section presents a way to adapt this representation model so as to make it sensitive to different motions.

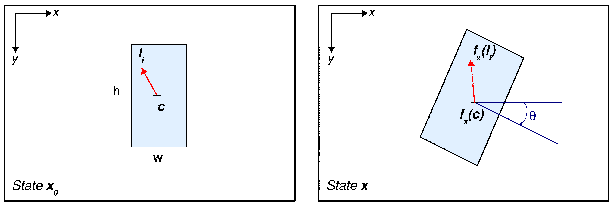


Fig. 1. Representation of the state to track and associated deformation from state $\mathbf{x}_0$ (left) to state $\mathbf{x}$ (right).

C. Multi-kernel configuration

As pointed out in [8], the use of several kernels (and thus several histograms) enlarges the measurement space, making the tracking more sensitive to different complex motions. A possible way to design a tracker sensitive to the motions considered here, is to design specific kernels for each motions. Indeed, examples of kernels sensitive to rotations are given in [8]. However, since several histograms need to be computed on the whole object, computational time is then increased, which makes actually such approach difficult to use in a particle filtering framework. The configuration proposed in this paper consists in using several identical kernels centered as illustrated in Figure 2. Weighting pixels locations with those kernels gives them a different importance according to their position in the object. In this case, the number of kernels do not lead to an increase in computational time since the histograms can be computed in the sub-parts of the object. Moreover, this kind of configuration can be used to detect a large range of motions.

Formally, a state $\mathbf{x}_k$ is associated to nine histograms $\{\mathbf{q}_j(\mathbf{x}_k)\}_{j=1..9}$ computed with the kernels $K_j(\boldsymbol{l} - \mathbf{c}_j)$. The distances $d_j(\mathbf{x}_k) = d(\mathbf{q}_j^*, \mathbf{q}_j(\mathbf{x}_k))$ are defined as in (4). Figure 3.b shows the shape of the distance function:

$$d_m(\mathbf{x}_k) = \frac{1}{9}\sum_{j=1}^{9} d_j(\mathbf{x}_k) \tag{5}$$

computed on the image of Figure 7-b. Multiple kernel representation being more discriminatory, the resulting distance

function varies faster than the one using single kernel representation. Therefore, the sensitivity of the tracker to motions like rotations is improved. More kernels can obviously be added in order to increase the tracking accuracy.

The next two sections presents two different ways to consider the problem of finding in each frame k the state $\mathbf{x}_k$ that best corresponds to the reference model, using color histograms as measurements.

III. Deterministic framework

This section shows how the formalism of [8] applies to our configuration, nine kernels being centered as illustrated in Figure 2.

Here, one histogram is computed with each kernel K_j using the n pixel locations of the object as in (2).

Following the general formalism described in [8], histograms are now considered in their vectorial form, $\mathbf{q} = (q_1{}^{(1)}, .., q_m{}^{(1)}, ..., q_1{}^{(9)}, .., q_m{}^{(9)})^\top$, where $q_u{}^{(j)}$ designs the value of bin u in the histogram computed with K_j.

To make optimization techniques possible, the distance in the histograms space for a candidate state $\mathbf{x}$ is considered here under the form:

$$O(\mathbf{x}) = \|\sqrt{\mathbf{q}^*} - \sqrt{\mathbf{q}(\mathbf{x})}\|^2 = 2 - 2\rho(\mathbf{x}) \tag{6}$$

where the square root is applied to each component of the vectors. Note that the minima of (6) correspond to the minima of (4).

The definition of each kernel function K_j is extended in the following to include the deformation $f_\mathbf{x}$ from state $\mathbf{x}_0$ to state $\mathbf{x}$ by defining for each pixel $\boldsymbol{l}$:

$$K_j(\boldsymbol{l}, \mathbf{c}_j, \mathbf{x}) = CK(f_\mathbf{x}(\boldsymbol{l}) - \mathbf{c}_j) \tag{7}$$

$$C = \frac{1}{\sum_i K(f_\mathbf{x}(\boldsymbol{l}_i) - \mathbf{c}_j)} \tag{8}$$

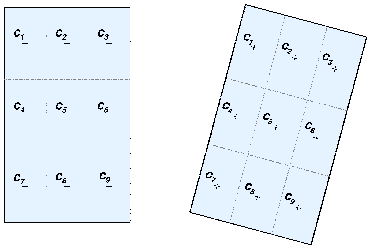
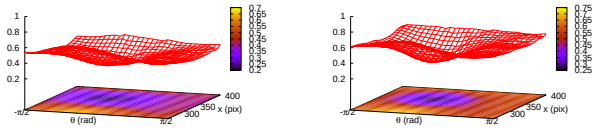


Fig. 2. Multiple kernels configuration.



(a) (b)

Fig. 3. Distance function w.r.t. rotation and x location, using simple (a) and multiple kernels (b) object representation.

where $\mathbf{c}_j$ is the center of kernel K_j.

The gradient of the kernel vector $\mathbf{K}_j(\mathbf{x}) = (K_j(\boldsymbol{l}_1, \mathbf{c}_j, \mathbf{x}), ..., K_j(\boldsymbol{l}_n, \mathbf{c}_j, \mathbf{x}))^\top$ is given by:

$$\mathbf{J}_{\mathbf{K}j} = \left[\frac{\partial \mathbf{K}_j}{\partial x}, \frac{\partial \mathbf{K}_j}{\partial y}, \frac{\partial \mathbf{K}_j}{\partial \theta}, \frac{\partial \mathbf{K}_j}{\partial s}\right]$$

The matrix $\mathbf{U}$ defined by $\mathbf{U} = \{u_{i,j}\}_{i=1..n, j=1..m}$ with $u_{i,j} = \delta_j(b(\boldsymbol{l}_i))$, links the n points with the color bin of the corresponding pixel in the considered frame (see [8]). Then, minimizing (6) using Newton's method leads to

$$\Delta\mathbf{x} = -2\mathbf{J}_\mathbf{U}^+\mathbf{e} \tag{9}$$

where $\mathbf{e} = \sqrt{\mathbf{q}(\mathbf{x})} - \sqrt{\mathbf{q}^*}$, $\mathbf{J}_\mathbf{U} = d(\mathbf{q}(\mathbf{x}))^{-\frac{1}{2}} \begin{bmatrix} \mathbf{U}^\top\mathbf{J}_{\mathbf{K}1} \\ ... \\ \mathbf{U}^\top\mathbf{J}_{\mathbf{K}9} \end{bmatrix}$, $\mathbf{J}_\mathbf{U}^+$ is the pseudo inverse of $\mathbf{J}_\mathbf{U}$, and $d(\mathbf{q}(\mathbf{x}))$ denotes the matrix with $\mathbf{q}$ on its diagonal.

In our work, Epanechnikov kernels have been used. For each pixel location $\boldsymbol{l}$:

$$K_j(\boldsymbol{l} - \mathbf{c}_j) = \begin{cases} C\left(1 - \frac{\|\boldsymbol{l}-\mathbf{c}_j\|^2}{h^2}\right) & \text{if } \|\frac{\boldsymbol{l}-\mathbf{c}_j}{h}\| \leqslant 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

h represents the span of the kernel.

Using (7), the kernel's dependency w.r.t. state parameters appears by writing that for each pixel location $\boldsymbol{l}$:

$$K_j(\boldsymbol{l}, \mathbf{c}_j, \mathbf{x}) = C\left(1 - \frac{\|f_\mathbf{x}(\boldsymbol{l}) - \mathbf{c}_j)\|^2}{h^2}\right)$$

where $f_\mathbf{x}(\boldsymbol{l})$ is defined in (1).

The optimization procedure allows good accuracy, but the tracking fails in case of occlusion or presence of several peaks in the cost function. In particular, for fast motions in which blurring (see Figure 4) makes the appearance of the object look uniform, the optimization is more likely to find a wrong minimum by shrinking down to a smaller portion of the object as shown in Figure 5.

The results obtained with this method are shown in the second row of Figure 7 for a moving object of fixed size.

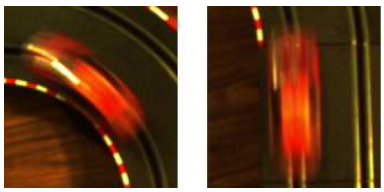


Fig. 4. Blurring due to object fast motion.

The next section presents particle filtering approach and how it can be adapted to our configuration.

IV. Particle filtering framework

A. Particle filtering principle

The use of Bayesian framework, and more particularly particle filtering [10] in vision applications, has been widely described in the literature . The main ideas are recalled here.

The question of estimating a state $\mathbf{x}_k$ given some observations $\mathbf{z}_{1:k}$ can be considered in an equivalent way as estimating the probability density function $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$, where $\mathbf{z}_{1:k}$ denotes $(\mathbf{z}_1, ..., \mathbf{z}_k)$. Assuming that the observations are independent and the system follows Markov rule leads to:

$$p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k}) \propto p(\mathbf{z}_k \,|\, \mathbf{x}_k) p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k-1}) \propto p(\mathbf{z}_k \,|\, \mathbf{x}_k) \int p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \,|\, \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}.$$

From this, the recursive steps of the non-linear optimal filter can be derived.

$$p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1}) \overset{evolution}{\longrightarrow} p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) \overset{update}{\longrightarrow} p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \tag{11}$$

In a Gaussian linear case, the above recursive filter admit an explicit expression given by the well-known Kalman filtering. However, the analytical solution remains unknown in the general case, and approximation methods have to be used. One possible approximation consists in linearizing the system to derive the extended Kalman filter. Although it has been shown sufficient in various applications, it is not constructed to handle the visual tracking tasks considered here, which may be non linear or with multiple modes.

The idea of particle filtering (also known as CONDENSATION [10]) is to approximate $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ by representing it by a finite set $\{s_k^{(i)}\}_{i=1}^N$ of N samples, or particles, associated with weights $\{w_k^{(i)}\}_{i=1}^N$:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \simeq p^N(\mathbf{x}_k \mid \mathbf{z}_{1:k}) \triangleq \sum_{i=1}^{N} w_k^{(i)} \delta_{s_k^{(i)}}(\mathbf{x}_k) \tag{12}$$

where $w_k^{(i)} \geq 0$ and $\sum_{i=1}^N w_k^{(i)} = 1$. Each particle thus represents a possible state of the object.

The recursive steps of the filter (11) above then lead to the following:

Knowing the set of N particles $\left\{(s_{k-1}^{(i)}, \frac{1}{N})\right\}_{i=1..N}$ at step $k-1$:

- **Evolution** of the particles according to a motion model, giving a new set : $\left\{(s_k^{'(i)}, \frac{1}{N})\right\}_{i=1..N}$.
- **Update:** Using the observation $\mathbf{z}_k$ the weight of each predicted particle is computed $w_k^{(i)} \propto p(\mathbf{z}_k \mid \mathbf{x}_k = s_k^{(i)})$. It gives the new set: $\left\{(s_k^{'(i)}, w_k^{(i)})\right\}_{i=1..N}$, with $\sum_{i=1}^N w_k^{(i)} = 1$.
- **Random weighted draw** of N particles from $\left\{(s_k^{'(i)}, w_k^{(i)})\right\}_{i=1..N}$, thus giving the new set: $\left\{(s_k^{(i)}, \frac{1}{N})\right\}_{i=1..N}$.

For initialization the particles are sampled on a Gaussian distribution around the initial known position. In the experiments a constant velocity model has been used for the evolution (or prediction) step. The particle filtering output considered is the estimator of probability expectation: $E[\mathbf{x}_k] = \frac{1}{N} \sum_{n=1}^{N} s_k^{(i)}$. The likelihood function is described in the following section.

B. Likelihood function

In particle filtering framework, the likelihood function needs to be evaluated for each particle and the set of particles itself gives a sampling of the probability density function to estimate. Therefore, no gradient needs to be computed as it is the case in optimization based methods such as [4] and [8]. In this section, *K* is set to a constant ensuring that the histograms are normalized. The spatial information is given by computing nine histograms in the nine parts defined in Figure 2 (and thus using one ninth of the n target's pixels) as in [5].

Formally, a state $\mathbf{x}_k$ is associated to the histograms $\{\mathbf{q}_j(\mathbf{x}_k)\}_{j=1..9}$. The likelihood of a state $\mathbf{x}_k$ is then defined by:

$$p(\mathbf{z}_k \mid \mathbf{x}_k) \propto exp(-\lambda {d_m}^2(\mathbf{x}_k)) \tag{13}$$

where d_m is given in (5) and λ is a constant parameter tuned empirically (a typical value is $\lambda = 20$).

Figure 5 presents comparative results between single kernel particle filter (1st row), multiple kernel deterministic approach (2nd row) and multiple kernel particle filter (last row) on a fast moving object sequence. As expected, the multiple kernel approach is more accurate than the single kernel one. In this sequence, the lack of information due to the blurring makes the optimization method fall into a wrong minimum (Figure 5 (2-c)), while the multiple kernel particle filtering allows good localization.

V. Combining both approaches

Up to this point, it is interesting to observe the complementarity aspects of the two considered approaches. In kernel-based optimization, a few iterations lead to the minimum of the cost function with good accuracy. However, the deterministic nature of the method makes it fail in case of complete occlusion. Furthermore, the algorithm can fall in a local minimum. In particle filtering, the density of probability is estimated without any assumption on its shape, allowing to keep several modes. Nevertheless, the accuracy depends linearly on the number of particles. A balance has to be found between increasing tracking accuracy and reducing computational time.

To keep the benefits of both algorithms, a way of combining them has to be considered. In [7], a mean shift embedded particle filtering is proposed, where mean shift optimization is applied to each particle to generate particles in local maxima of the probability density function. It allows to use fewer particles with good accuracy. However, in the multiple kernel tracking case, computing the optimization procedure on each particle would be time consuming. Therefore, we choose to apply the optimization directly to the estimate of the particle filter. The decrease in the number of particles introduces a loss of accuracy that is compensated by the optimization methods which finds the mode of the distance function. Of course the particle filter estimate has to be close enough to this mode, but experiments show that it can be achieved with a small number of particles.

To avoid a failure due to temporary occlusions, a threshold on the distance function allows to detect them and switch to the particle filter estimate. If $\overline{\mathbf{x}}_k{}^{PF}$, and $\overline{\mathbf{x}}_k{}^{PFSSD}$ denotes respectively the estimate given by the particle filter and

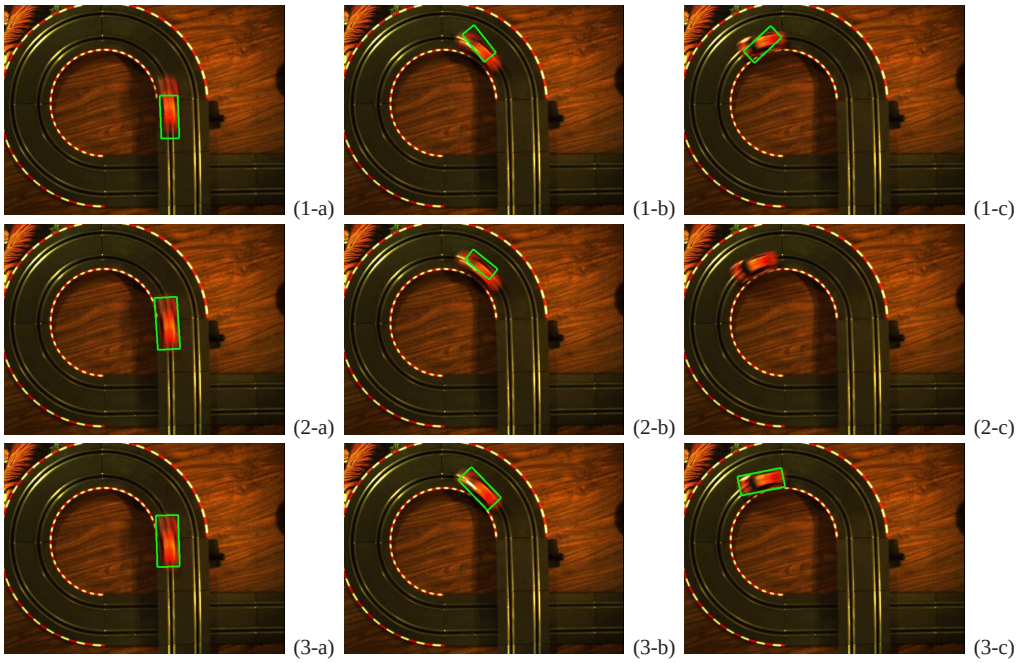


Fig. 5. Comparative results of the different approaches with translations, rotations and scale. First row: particle filter with 1 kernel, using 500 particles. 2nd row: Multiple kernels deterministic approach [8]. 3rd row: particle filter with 9 constant kernels, using 500 particles.

the combined algorithm in frame k, the procedure can be formulated as follows:

if $d_m\left(\overline{\mathbf{x}}_k{}^{PF}\right) \geq \epsilon$ **then**
  compute $\overline{\mathbf{x}}_k{}^{PFSSD}$ from $\overline{\mathbf{x}}_k{}^{PF}$ using optimization
else
  $\overline{\mathbf{x}}_k{}^{PFSSD} \leftarrow \overline{\mathbf{x}}_k{}^{PF}$
end if

ϵ being a threshold to be tuned empirically.

While the optimization method alone can fail in case of large motions, combining it with particle filtering makes it take advantage of the estimate as a prediction close to the object state. Furthermore, thanks to the prediction, it reduces the number of iterations required in the Newton minimization.

VI. TRACKING FOR SERVOING APPLICATION

The tracking algorithms described in the previous sections provide us with the center of gravity, size and orientation of the object in the image.

A. Tracking

The comparative results of different approaches on a sequence with total occlusion and fast motion with fixed size are given in Figure 6 and 7. A particle filter using a single kernel is tested, giving poor results in estimating orientation of the object (see Figure 6 (1) and first row of Figure 7). As expected, the optimization method is accurate but fails when confronted to the occlusion (Figure 7 (2-c), corresponding to rupture in frame 46 of Figure 6 (2)). The particle filter with 75 particles ((4) in Figures 7 and 6) is robust to the occlusion but is less precise than using 500 particles (3). The experimental results show that the combined algorithm (5) reaches the same accuracy as the particle filter alone (3),

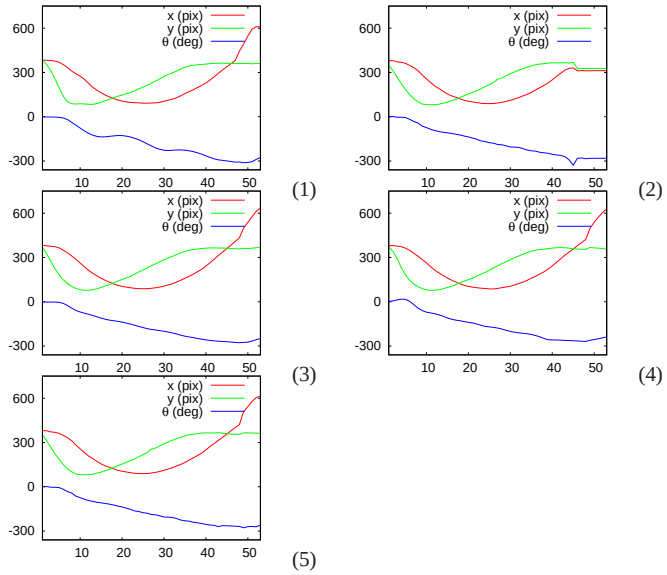


Fig. 6. Position and orientation of the object in the sequence of Figure 7 as obtained using: (1) single kernel particle filtering with 500 particles, (2) multiple kernels deterministic approach [8], (3) multiple kernel particle filtering with 500 particles, (4) multiple kernel particle filtering with 75 particles, (5) combined algorithm with 75 particles. Abscissa indicates the frame number.

using much fewer particles, leading to a far more efficient approach for a computational time point of view since it is about 3 times faster.

B. Image-based visual servoing

This section describes how these elements can be used to design a control scheme to servo a robot to a moving object.

The aim of the image-based visual servoing is to minimize the error **e** between the current value of a set of *k* visual

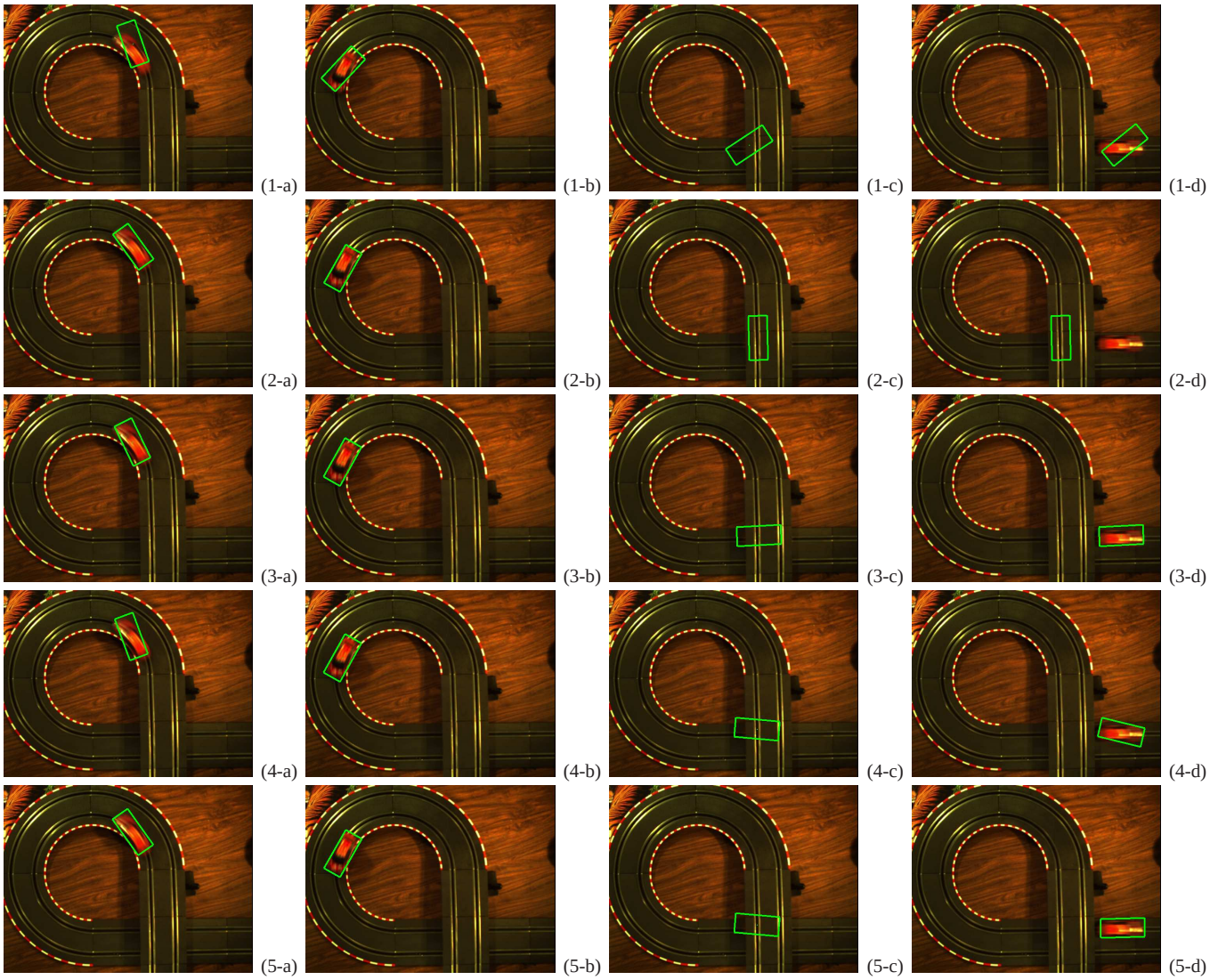


Fig. 7. Comparative results of the different approaches with translations and rotations. First row: single kernel particle filtering, using 500 particles. 2nd row: multiple kernels deterministic approach [8]. 3rd row: multiple kernel particle filtering, using 500 particles. 4th row: multiple kernel particle filtering, using 75 particles. 5th row: combined algorithm, using 75 particles.

features $\mathbf{s}$ and its desired value $\mathbf{s}^*$:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{14}$$

Let $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega})$ denote the camera instantaneous velocity, with $\boldsymbol{v}$ being the linear velocity of the origin of the camera and $\boldsymbol{\omega}$ the angular velocity of the camera frame. In the case of a moving object, the time variation of this error is given by:

$$\dot{\mathbf{e}} = \dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v} + \frac{\partial \mathbf{e}}{\partial t} \tag{15}$$

where $\mathbf{L_s} \in \mathbb{R}^{k\times 6}$ is the interaction matrix related to $\mathbf{s}$. The term $\frac{\partial \mathbf{e}}{\partial t}$ represents the time variation of $\mathbf{e}$ due to the object self-motion.

C. Visual features and interaction matrix

From the center of gravity coordinates (x, y), size and orientation given by the trackers presented, we can easily derive four visual features. To get a well-conditioned interaction matrix, the following features are used in the experiments:

$$\mathbf{s} = (x_n, y_n, a_n, \theta)^\top \tag{16}$$

where $a_n = \frac{1}{\sqrt{A}}, x_n = a_n x, y_n = a_n y$ and θ is the orientation defined in section 2. The interaction matrices of these features have been derived in [12] using moments.

D. Estimating self-motion of the object

To ensure an exponential decrease of $\mathbf{e}$ (that is $\dot{\mathbf{e}} = -\lambda\mathbf{e}$), the following control law is obtained from (15):

$$\mathbf{v} = -\lambda\widehat{\mathbf{L}_\mathbf{s}^+}\mathbf{e} - \widehat{\mathbf{L}_\mathbf{s}^+}\widehat{\frac{\partial \mathbf{e}}{\partial t}}. \tag{17}$$

To avoid tracking errors, a good estimation of $\frac{\partial \mathbf{e}}{\partial t}$ is thus needed. From (15) a first approximation of $\frac{\partial \mathbf{e}}{\partial t}$ is:

$$\widehat{\frac{\partial \mathbf{e}}{\partial t}} = \widehat{\dot{\mathbf{e}}} - \widehat{\mathbf{L}_\mathbf{s}}\widehat{\mathbf{v}} \tag{18}$$

where $\widehat{\dot{\mathbf{e}}}$ can be approximated in discretization by:

$$\widehat{\dot{\mathbf{e}}}_k = \frac{\mathbf{e}_k - \mathbf{e}_{k-1}}{\Delta t}. \tag{19}$$

For the experiments, Kalman filtering on $\widehat{\frac{\partial \mathbf{e}}{\partial t}}$ has been implemented for the experiments, with a constant velocity

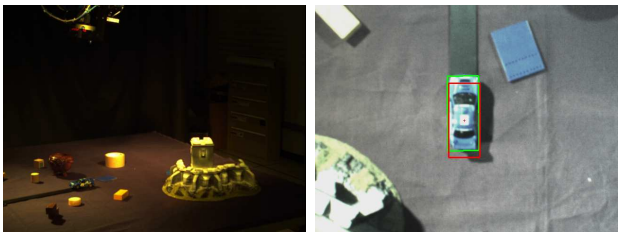


Fig. 8. Tracking a moving object.

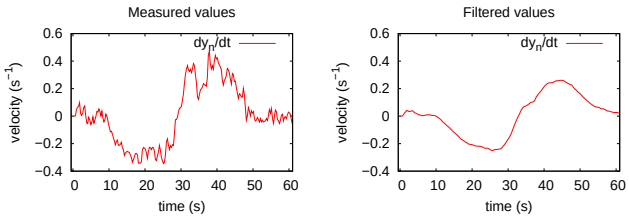


Fig. 9. Measured and estimated velocity of the y_n visual feature due to the object self-motion.

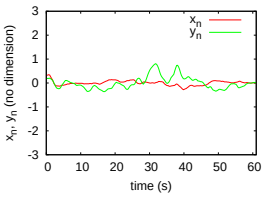


Fig. 10. Error of the x_n and y_n features. Desired value is zero.

state model, using (18) and (19) for the measure (as in [13]). It allows reducing tracking error (see Figure 10) due to object self-motion.

E. Experimental results

The method presented above has been tested for tracking a moving object (the blue car in Figure 8) using a camera mounted on the end effector of 6 degrees of freedom gantry robot (Figure 8 left). The desired position is shown by the red rectangle. The green one represents the current position of the car as estimated through the tracking algorithm. The video of this experiment is provided with this paper. Since the car is moved manually, there is no ground truth of the motion. However, the main component of the motion is the translation on the y axis, in one direction then the other. As expected, the sign of the estimated velocity of the object on this axis changes when the direction of the object motion changes (see Figure 9). The use of an estimator of the velocity allows to keep a small error during the motions. Since the object self-motion is estimated with a constant velocity model, the brutal variations of the object velocity introduce a tracking error. Indeed, the velocity changes appearing in frame 30 and 37 (see Figure 9) results in an increasing error as it can be observed in Figures 10 and 11c.

VII. CONCLUSIONS AND FUTURE WORKS

To track moving objects, whose appearance may vary during motion, density-based object representations such as color-based histograms are an attractive choice. When dealing with complex motion however, single kernel methods are no more efficient. The use of a multiple kernel representation

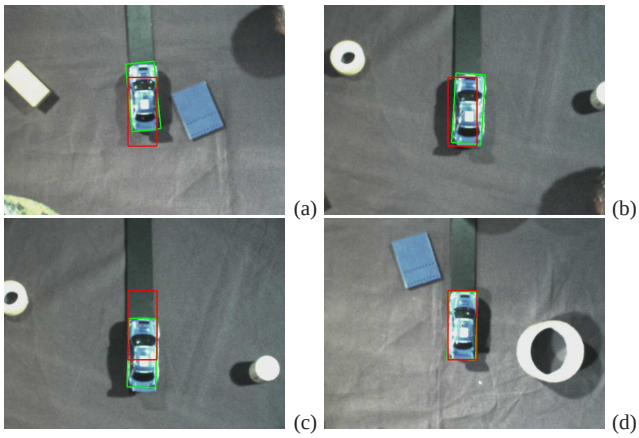


Fig. 11. Internal view. Red rectangle represents the desired position, the green one is the current position as given by the tracking algorithm. (a) and (b) shows examples of view with small rotation and translations errors. (c) corresponds to the direction change, which induces a tracking error. (d) is the final position. Although the target seems to be vertical, let us recall that the camera is controlled in order to maintain this position in the image.

of the object is then necessary to make the tracking sensitive to a large range of motions. By combining multiple kernel optimization and particle filtering approaches, we proposed a robust tracking algorithm which allows to estimate the position, orientation and size of a moving object. The proposed approach has proved to be able to handle fast motions using roughly 75% fewer particles. Without any particular optimization, we used the proposed approach within a visual servoing experiment at a slow rate (5 fps). Optimizing computational time as well as the use of a more precise evolution model will allow to reduce even more the tracking errors.

REFERENCES

[1] G. Hager, P. Belhumeur, "Efficient Region Tracking With Parametric Models of Geometry and Illumination", *IEEE T. on PAMI*, vol. 20(10), pp. 1025-1039, 1998.

[2] F. Jurie, M. Dhome. "Hyperplane Approximation for Template Matching", *IEEE T. on PAMI*, vol. 24(7), pp. 996-1000, 2002.

[3] E. Malis, S. Benhimane. "Homography-based 2D Visual Tracking and Servoing", *IJRR*, vol. 26(7), pp. 661-676, 2007.

[4] D. Comaniciu, V. Ramesh, P. Meer, "Real-Time tracking of non-rigid objects using mean shift',' *IEEE CVPR*, pp. 142-149, May 2000.

[5] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, "Color-Based Probabilistic Tracking", *ECCV'02*, pp. 661-675, May 2002.

[6] K. Nummiaro, E. Loller-Meier, L. Van Gool. "An adaptive color-based particle filter", *Image and Vision Comp.*, vol.21, pp. 99-110, 2003.

[7] C. Shan, T. Tan, Y. Wei, "Real-time hand tracking using a mean shift embedded particle filter", *Pattern Recognition*, vol. 40, pp. 1958-1970, 2007.

[8] G. Hager, M. Dewan, C. Stewart, "Multiple kernel tracking with SSD", *Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 790-797, 2004.

[9] R. Megret, M. Mikram, Y. Berthoumieu, "Inverse Composition for Multi-kernel Tracking", *Computer Vision, Graphics, and Image Processing, LNCS 4338, Springer*, pp. 480-491, 2006.

[10] M. Isard, A. Blake, "CONDENSATION: conditional density propagation for visual tracking", *Int. Journal of Computer Vision*, vol. 29, pp. 5-28, 1998.

[11] Z. Zivkovic, B. Krose, "An EM-like algorithm for color-histogram-based object tracking", *IEEE CVPR*, vol.1, pp. 798-803, 2004.

[12] O. Tahri, F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects", *IEEE T. on Robotics*, vol. 21(6), pp. 1116-1127, Dec. 2005.

[13] F. Chaumette, A. Santos, "Tracking a moving object by visual servoing," *12th IFAC World Congress*, vol. 3, pp. 643-648, Jul. 1993.